

Assignment 3: Getting Answers

Overview

In this assignment, we continue to construct a natural language query system.¹ In the first assignment, we wrote a procedure to *match* questions typed by a user, called **source**, with **patterns** that we define. This *match* procedure will allow our system to identify what the user is asking. In the second step, we wrote code to import some data to find the **answers** to the users' questions. In this assignment, we'll write code that connects **patterns** with action functions that empower the system to get the **answers**.

More Details

Notice in the a3.py starter code, we have this line:

```
from match import match.
```

This brings the *match* function into our code so that we can use it, just like any other function, e.g. `match(["an", "_", "example", "pattern"], ["an", "awesome", "example", "pattern"])`

Additionally, notice this line:

```
from data import features
```

This brings the *features* dictionary into our code so that we can use it, just like any other variable, e.g.

```
print("The area of China is: ", features["area"]["China"][1])
```

¹ The original version of this assignment is taken from the textbook Concrete Abstractions: An Introduction to Computer Science Using Scheme, by Max Hailperin, Barbara Kaiser, and Karl Knight, Copyright (c) 1998 by the authors. Full text is available for free [here](#). The assignment evolved at Pomona College through several instructors' offerings, with changes by Nathan Shelly and Sara Sood, Northwestern University.

Your job

1. Implement the outlined set of action functions. These functions have docstrings and asserts in *a3.py* that demonstrate their intended behavior. This includes the following:

- `country_by_rank`
- `rank_by_country`
- `list_countries`
- `list_patterns`

ALL of these action functions take a list of strings as input. You can think of this input list as the output of a call to `match`. For example, if we made the following call to `match`,

```
match(["which", "country", "is", "ranked", "number", "_", "for", "%"],  
      ["which", "country", "is", "ranked", "number", "2", "for", "population"])
```

We should get back `['2', 'population']`. That is, the source matched the pattern, and '2' and 'population' are the items that are substituted for the `_` and `%`.

`['2', 'population']` will (after we implement the `search_pa_list` function) get passed as input to the `country_by_rank` function.

```
country_by_rank(['2', 'population'])
```

We would expect this to return `['india']`.

2. Implement `search_pa_list` (given the provided docstring and asserts)

3. Write a `query_loop` procedure to manage interactions with the user. See provided docstring.

3. Come up with 3 new action functions and at least 3 new pattern/action pairs (add to `pa_list`) that utilize the actions. Be creative here. You could write simple actions, like the ones provided, or more complicated ones that involve utilizing data from multiple features.

Demonstrate usage of each pattern/action pair by writing a new assert for the `'search_pa_list'` function that shows the new questions that the user can ask.

For example, the following assert demonstrates the `country_by_rank` function:

```
assert search_pa_list(["which", "country", "is", "ranked", "number", "2", "for",  
                      "median", "age"]) == ["japan"], "search_pa_list test 2"
```

Some assert statements are provided in *a3.py* but you should write more asserts (in addition to interacting with your system as a user) to properly test your code.

4. So that we can easily grade your systems and added pattern/action pairs, please copy and paste a transcript demonstrating interactions with your chatbot to the end of your code file (be

sure to comment it out). This transcript should include any additional questions/patterns that you created.

Complete your work in Assignment3.py and upload it to canvas. You do not need to upload any of the other files provided; we have copies for grading.